University of Oxford

Department of Statistics

Handbook for Further LaTeX

Michaelmas Term 2014

Susan Hutchinson

Computing support specialist

Last changed on 15-11-2014 at 17:21 by Susan Hutchinson

Revision 1.240

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Professor Brian Ripley for his permission to use material from exercises he devised for the Department of Statistics as part of an introductory LaTeX course.

I have also made use of ideas from Dr Nicola Talbot's excellent documents *LaTeX for Complete Novices* and *Using LaTeX to write a PhD Thesis*. These are exceptionally clear and relevant and can be found at `http://www.dickimaw-books.com/latexresources.html`.

Both have probably forgotten more LaTeX than I will ever know.

Chapter 1

Introduction

This handbook contains the following documents:

- A set of exercises exploring different LaTeX features.
- The answers to the exercises.
- The slides.

Chapter 2

Before you start

2.1 Downloading files

Follow these commands to download some sample files which you will need for the exercises.

1. Create a new folder
2. Browse to `http://www.stats.ox.ac.uk/pub/susan/oucs_latex/samples.zip` to download a bundle of useful files that you will need today. I have also included a copy of this document as some of the features are only obvious when viewed online.
3. Right click on `samples.zip` and select **Extract All...** to extract all the files.

2.2 Texmaker

We will be using Texmaker to create, edit, compile and view LaTeX files. To start Texmaker go to Start ▶ All Programs ▶ Texmaker. A window like Figure 2.1 should appear:

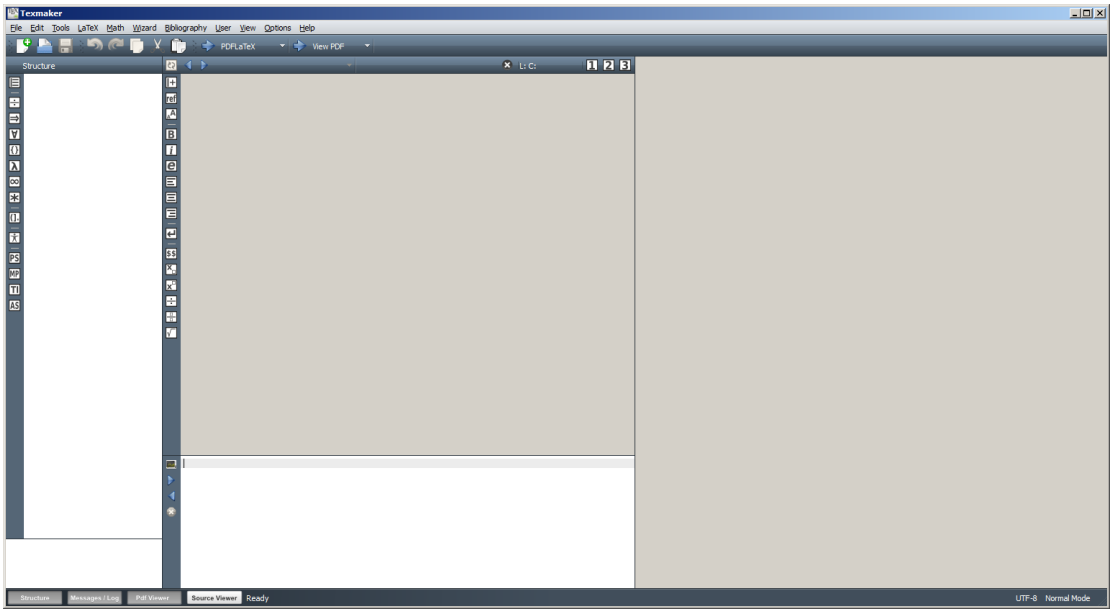

Figure 2.1: The Texmaker: Start–up screen

Take a little time to explore Texmaker. There are several windows below the menus buttons. They are:

Structure window The narrow window on the lefthand side is used to help you find your way around larger projects. The small window below this is used to show file names.

Edit window The large window is where files are edited.

Output window The thin window at the bottom is where the output from compilation appears. This is where you should look when you have problems.

Display window The large window on the right is used to display the compiled version of the file.

If you cannot see all these windows use the **View** menu to find them.

Notice the large arrows followed by **PDFLaTeX** and **View PDF** below the menu bar. Clicking the arrows will compile and view your LaTeX file.

Find the following buttons and short cuts:

- **Run (F6)** This is the same as clicking on the **PDFLaTeX** arrow.
- **View Output (F7)** The same as clicking on the **View PDF** arrow.
- **Find (Ctrl+F)**
- **Replace (Ctrl+R)**
- **Save (Ctrl+S)**

The keyboard shortcuts are included after each command. The buttons are used for compiling, viewing, saving and editing LaTeX documents.

Most of the other buttons are used for typesetting. See if you can find some of these:

- **Bold**
- ***Italic***
- **Align left**
- **Itemization**
- **$$** Inline maths

Now you have explored the Texmaker it is time to start using LaTeX.

2.3 LaTeX

The exercises in this document are only suggestions. If there are other packages and techniques that you want to explore then please do so and we will do our best to assist you but priority will be given to those needing help with these exercises.

Most sets of exercises include a "If you have time" section. These are designed to develop a particular technique. Not all the information needed to complete the exercises is necessarily included – you may need to search online or read the documentation.

2.4 Typesetting conventions

Distinct fonts and structures have been used to distinguish between various LaTeX features.

i. Text that needs to be typeset exactly as it appears in the exercises looks like this:

```
\begin{document}
\usepackage{parskip}
\begin{document}
Hello.
\end{document}
```

[For the curious this is achieved adding the `\usepackage{Verbatim}` to the preamble and enclosing the text in a `Verbatim` environment.]

ii. Texmaker output profiles and other commands appear in a small sans-serif font like this: **PDFLaTeX**.

iii. The output from a LaTeX compilation which appears in the display window I have enclosed between two lines as follows:

Here's a reference to Figure 2.1 in Chapter 2.

Chapter 3

Getting Started

The first exercise is a refresher to get you started.

i. Start Texmaker and create a new file (**File** $->$ **New**).

ii. Save the file as exercise1.tex.

Note that if you prefer to use TEXworks or Wordpad/Notepad and the Command Prompt window then do so.

▷ Exercise 1 **Creating a sample file**

Insert the following lines into your new file and then save it.

```
\documentclass[a4paper,12pt]{report}
\title{My sample document}
\author{My Name}
\date{November 2014}
\begin{document}
\maketitle
\pagenumbering{roman}
\tableofcontents
\chapter*{Acknowledgements}
\pagenumbering{arabic}
\chapter{Introduction}
\chapter{Background}
\chapter{Recent developments}
\chapter{Areas for study}
\chapter{Conclusion}
\end{document}
```

Most of the markup in the document should be familiar but please ask if any of the commands are not clear.

Now run **PDFLaTeX** to compile the file by clicking the blue arrow to the left of **PDFLaTeX**. To see the results of the compilation click the blue arrow to the left of **View PDF**. There should be no errors, but there's not a lot of useful information here, either. We will develop this document as the course progresses. Remember, too, that you will need to compile the document twice in order to see the information in the table of contents. When done your Texmaker screen should look like Figure 3.1.

Note that the structure of the document is shown in the window to the left of the central editing window. This will be useful when editing multi-file documents.

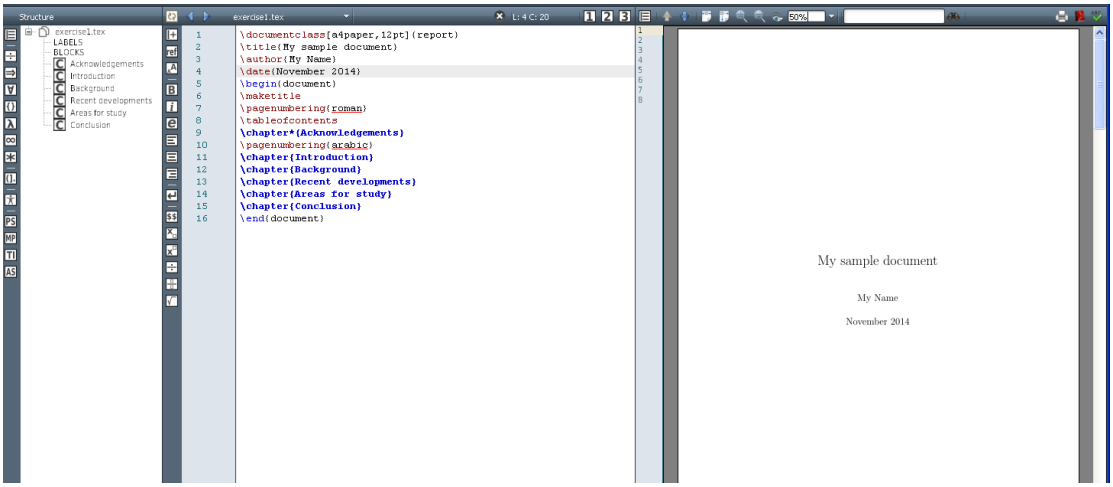

Figure 3.1: Exercise 1 compiled

From now on I won't ask you to click on the run button and view button but just say recompile your LaTeX file.

▷ Exercise 2 **Changing the document**

1. Add your name and a title to the title page.
2. Add some text to the Introduction chapter. Perhaps you could write a couple of sentences about your studies or a hobby.

When you've saved the changes, recompile the file. The preview window – in this case Adobe Reader – will reload automatically with the new version after the compilation has completed successfully.

Before we go on to the next set of exercises make sure you are confident with the edit → compile → preview cycle. You will be doing this many times during the course.

Chapter 4

Long documents

It is easier to manage longer documents like a thesis or a dissertation if they are organised in separate files. An advantage of this approach is that certain chapters only can be selected for compilation. For example, a chapter with lots of pictures can be omitted to speed up the edit → compile → preview cycle while writing other parts of the document. If this file containing the chapter has previously been compiled then references to figures in that chapter will still work.

▷ Exercise 3 **Splitting up a large document**

Make a backup copy of your project. Now we are going to remove all the text associated with the first chapter and put it in a separate file. Of course the words you added in the previous exercise may well be different from mine!

1. Copy the lines

   ```
   \chapter{Introduction}
   Here are some words for the first chapter.

   There's not a lot but it's a start.
   ```

   into a new file and save it as `intro.tex`. If you look at in the bottom lefthand corner of the Texmaker screen you should see the names of your two files as in Figure 4.1.

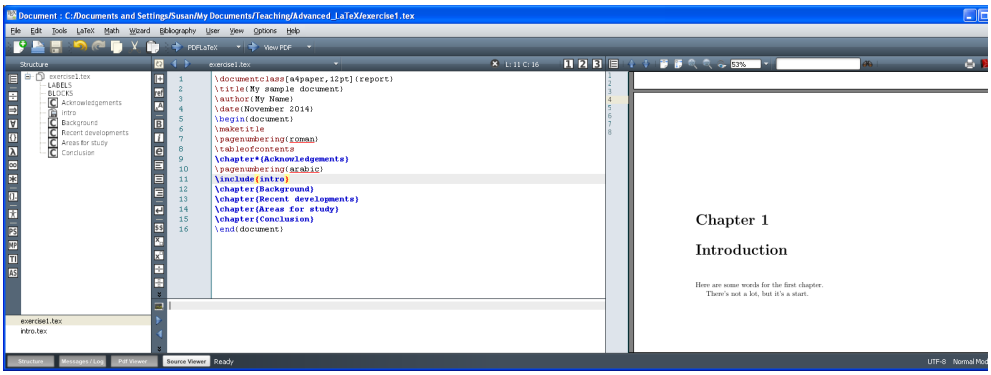

Figure 4.1: Texmaker window showing two file names

2. Delete these lines from your main file, `exercise1.tex`.

3. Add the line `\include{intro}` to `exercise1.tex`.

4. Recompile `exercise.tex` and preview it.

   Make sure that you always compile your main file; compiling `intro.tex` *will produce many errors and will fail.*

It should look exactly the same as the original version.

Now replace `\chapter{Background}` with `\include{background}` and create a new file called `background.tex` with the first line `\chapter{Background}` and recompile. Again, after compilation the document should look exactly the same.

▷ Exercise 4 **Conditional compilation**

We are now going to test that references will work even if we leave out some chapter files. Now make these changes:

1. Add these lines to `background.tex`

   ```
   \label{ch:background}
   I'm going to add some words and a picture in the second chapter.
   \begin{figure}[ht]
   \centering
   \includegraphics{ox-crest.png}
   \caption{The University of Oxford crest}
   \label{fig:OxCrest}
   \end{figure}
   ```

2. Make sure that the file `ox-crest.png` that you downloaded earlier is in the same folder as your `.tex` files.

3. Add `\usepackage{graphicx}` to the preamble in the main file, `exercise1.tex`.

4. Add these lines to `intro.tex`

   ```
   Here's a reference to Figure~\ref{fig:OxCrest} in
   Chapter~\ref{ch:background}.
   ```

   and recompile *twice* and preview the output.

You should see something like this in your preview window.

Here's a reference to Figure 2.1 in Chapter 2.

Now we are going to run a conditional compilation, including only `intro.tex` and excluding `background.tex`.

1. Add the line `\includeonly{intro}` to the preamble of the main `.tex` file.

2. Save the file and recompile.

The new document will be missing Chapter 2 but the references will still be correct and there will still be an entry labelled 'Background' in the table of contents.

If you have time…

If you have time put all the information in the preamble in a separate file. See if you can explain how the `\input{file.tex}` command differs from `\include{file.tex}`. You may find google useful. Would you use `\input{file.tex}` or `\include{file.tex}` for the information in the preamble?

Chapter 10 contains the answers.

Chapter 5

Customising LaTeX

Often you will find that you are repeating the same LaTeX markup over and over. The next exercise demonstrates how to create your own commands which can then be reused throughout your document.

For example if we have many occurrences of the phrase 'Oxford University' in our document it is much quicker to add the line `\newcommand{\OU}{Oxford University}` to the preamble. Subsequently, whenever we want to write 'Oxford University' all that is needed is to include the markup `\OU` in the text.

▷ Exercise 5 **Creating a new command**

Make a backup copy of your project. We are now going to add the new command to the preamble in the main file and make use it in both this document and in `intro.tex`.

You may want to remove or comment out (using a %) the `\includeonly...` line.

1. Add `\newcommand{\OU}{Oxford University}` to the preamble.
2. Add a suitable sentence such as 'I am grateful to `\OU` for the support I have received' to the Acknowledgements chapter. You may need to add { } after `\OU` to make sure the spacing is correct.
3. Add another sentence including `\OU` to your `intro.tex`.
4. Save all the files and recompile. Open a preview window and check that you can see 'Oxford University' in the text.
5. At this point you realise that it should be 'University of Oxford' not 'Oxford University'. Change the `\newcommand{\OU}...` line in your main file and recompile.

This demonstrates the advantage of using `\newcommand` for frequently used phrases or commands. If you need to make a change, it is only necessary to do it in one place – that is in the preamble.

▷ Exercise 6 **Using arguments with new commands**

Let's assume our document is going to make reference to several different Universities. Whenever we do that we want to change the font to bold, and make the colour magenta.

1. We need to add `\usepackage{xcolor}` to the preamble.
2. Add a line

   ```
   \newcommand{\Uni}[1]{\textbf {\color{magenta} University of #1}}
   ```

   to the preamble.

3. Now add two or more entries to your text using the `\Uni` command. Perhaps you could have `\Uni{Manchester}` and `\Uni{Cambridge}`.
4. See if you can make the font bold and sans serif and change the colour to blue. You will need to add the `\textsf` command.

Chapter 10 on page 21 contains answers where they don't appear in the text.

▷ Exercise 7 Customising existing commands

It is also possible to modify existing commands so that they behave as you want. The classic example of this is the `\today` command which by default produces:

November 15, 2014

---

which is an American format. Many people prefer a date of the form 15 November 2014.

The standard definition of `\today` is

```
\newcommand{\today}{\ifcase\month\or
  January\or February\or March\or April\or May\or June\or
  July\or August\or September\or October\or November\or December\fi
  \space\number\day, \number\year}
```

Using `\renewcommand` see if you can change the format of the `\today` command. You will need to add this to the preamble. See chapter 10 for the answer.

If you have time…

See if you can make the colour of your new `\Uni` command an argument as well so that the command is `\Uni{green}{Leeds}` for example. Again, see Chapter 10 for the answer.

Chapter 6

Creating a slide show

If you are writing papers or other technical documents in LaTeX you will probably need to give a presentation at some point. The decision to use a LaTeX slide–making environment is often based on the same considerations: typesetting maths is straightforward, creating precisely your own customisations is possible.

There are several slide-making document classes available, including the original class `slides` and the more powerful `Prosper`. One of the most popular and powerful classes is `beamer`. Early distributions of LaTeX did not always include Beamer, but up-to-date disbributions do.

Using Beamer it is possible to create complex and sophisticated slideshows which include equations, pictures and graphics, bibliographies, sound and animations. Effects such as pauses and transitions are simple to create; familiar LaTeX commands work.

▷ Exercise 8 **Creating a slide show**

i. Creating a minimal slide show based on the example below or use the example file provided. A very simple beamer `.tex` file could look like this:

```
\documentclass[pdf]{beamer}
% Put packages here
\usetheme{Warsaw} % Choose a theme
\usecolortheme{rose}  % Choose a colour scheme
\title{My first slideshow}
\subtitle{I hope you like it}
\author{Susan Hutchinson}
\institute{University of Oxford}
\date{March 2010}
% End of preamble
\begin{document}
\maketitle
\begin{frame}
\frametitle{Outline}
\tableofcontents
\end{frame}
\section{Introduction}
\begin{frame}
\frametitle{My first slide}
Contains very little information
```

```
\end{frame}
\end{document}
```

- Save this in a new project called `slideshow` and compile it with **LaTeX** => **PDF**.
- Open the PDF file - and Adobe Acrobat should start - then select **Window ▶ Full Screen Mode**. You should now be able to use the arrow or **Page Up** and **Page Down** keys to display the slides.

ii. Develop your slide show by changing themes and colours. The lines

```
\usetheme{Warsaw}
\usecolortheme{rose}
```

are used to set up the theme and colour scheme. Change and recompile your slide show a couple of times with different themes and colours. Do not spend too long on this. Most of the installed themes and colours are listed in the Tables 6.1 and 6.2.

AnnArbor	Antibes	Bergen	Berkeley	Berlin	Boadilla
CambridgeUS	Copenhagen	Darmstadt	Dresden	Frankfurt	Goettingen
Hannover	Ilmenau	JuanLesPins	Luebeck	Marburg	Madrid
PaloAlto	Pittsburgh	Rochester	Singapore	Szeged	Warsaw

Table 6.1: Some Beamer themes

albatross	beaver	beetle	crane	dolphin	dove	fly
lily	orchid	rose	seagull	seahorse	whale	wolverine

Table 6.2: Some Beamer colours

iii. Make the following changes to your slide show

- Add some more content so that there are three or four slides
- Add some maths.
- Add pauses and overlays; the slides contain some examples of the markup for pauses and overlays.
- Add at least one more `\section` command before a frame so that the contents of the second slide is changed. Remember that you will need to compile twice in order for new sections to be displayed.

iv. Explore the examples provided and the package documentation in the `beamer` package folder. This can be found at `C:\Program Files\MiKTeX 2.7\tex\latex\beamer`.

v. Using the `beamer` documentation and Google see if you can work out how to create a handout rather than a presentation. In the handout version slides with pauses and overlays appear as one slide with all the information, rather than a series of slides.

If you have time...

Using the `beamer` documentation and Google if you get stuck, see if you can add a movie to your slide show. You can download a movie file from `http://www.stats.ox.ac.uk/pub/susan/oucs_latex/movie.avi` . Apologies for the poor quality and to those who don't enjoy cycling!

See Chapter 10 for a suggested solution.

Chapter 7

Exploring packages

Many packages are available in LaTeX. These add features or change the behaviour of existing packages. We will be looking at just a few commonly used packages in this set of exercises.

Make a backup copy of your project.

▷ Exercise 9 The `fancyhdr` package

The basic headers provided by LaTeX are rather limited. The `fancyhdr` packages adds more information in the header and footer and extends the customistations you can make, allowing you, for example, to add your name to the footer of each page.

1. Add `\usepackage{fancyhdr}` to the preamble.
2. Add `\pagestyle{fancy}` just after the `\maketitle`.
3. Compile your main file and preview it. Do you notice any difference?

It could be that you can't see any difference to your document. If each chapter is only a page long then nothing will have changed. This is because header information is only added to the second and subsequent pages of any chapter. Add something the following to a chapter:

```
\newpage
With a second page. It may be that headings
only appear on the second
page of a chapter.
\section{My big idea}
Here's the second page.

With a second paragraph.
\newpage
Again we need a new page to see what the
default headings look like.
\subsection{More detail}
Why this is an excellent idea.
\newpage
More information about my very good idea.
```

and recompile (possibly a couple of times if you want references to be correct). Now look at the document. You should notice

at the top of the second page. If you have included sections as in the example above you should see something like this.

Notice how the titles and numbers of the chapter and section now appear in the header. We are now going to customise the `fancyhdr` package by including our name in the footer.

▷ Exercise 10 Using the `texdoc` command

LaTeX packages mostly come with documentation. Some LaTeX editors provide this documentation as part of their help pages, others don't. We are now going to read the documentation for `fancyhdr` outside TeXnicCenter. We will use a command prompt and the `texdoc` command.

i. Open a command prompt window from **start ▸ All Programs ▸ Accessories ▸ Command Prompt**. You should see a window like Figure 7.1.

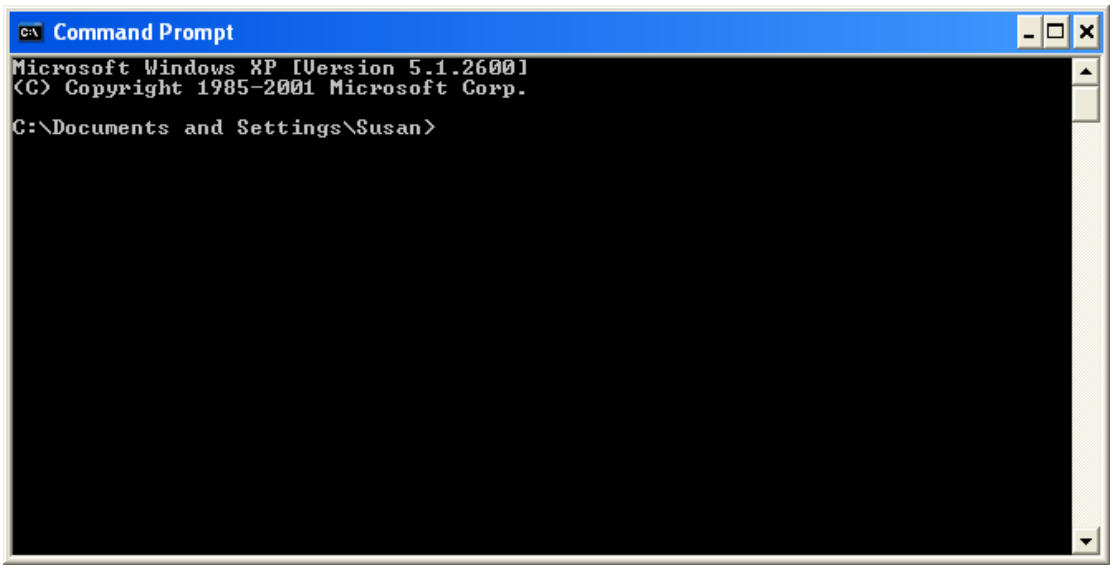

Figure 7.1: The command prompt window

ii. Type in the command `texdoc fancyhdr` and press enter. You may need to click on a link in a browser window but you should finally see a page like that in Figure 7.2:

From this document see if you can find out how to make the following changes to the footer:

- Add your name to the lefthand side of the footer.
- Move the page number from the centre to the righthand side.

▷ Exercise 11 Creating a glossary

I suggest you look at `finalglos.pdf` to see how your final version should look.

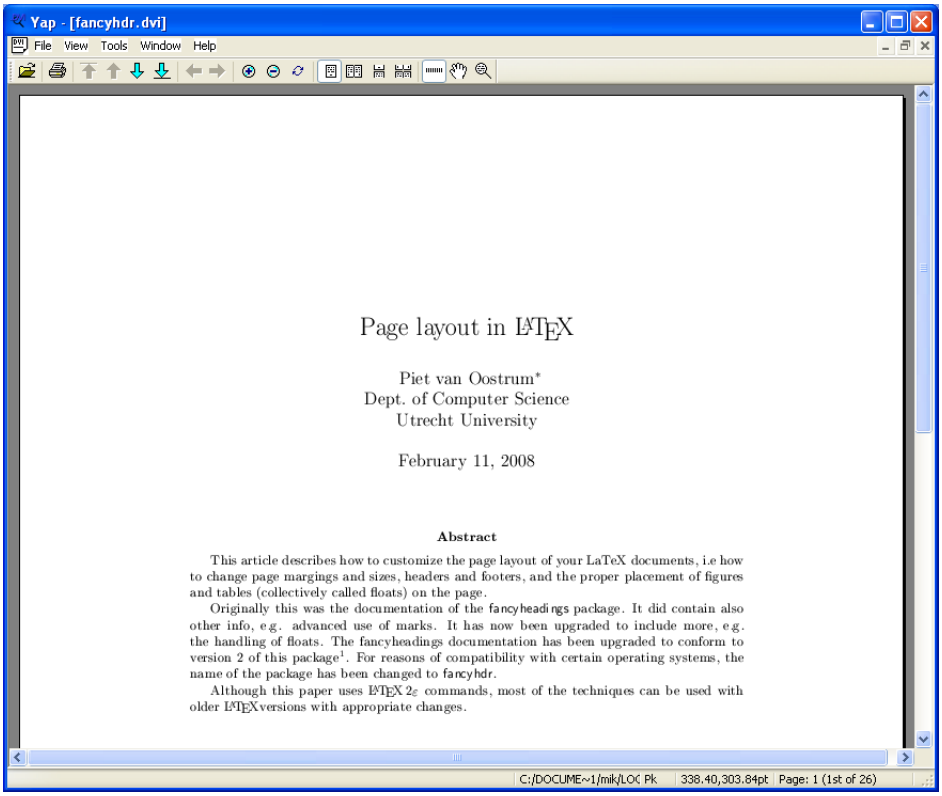

Figure 7.2: fancyhdr documentation

A glossary allows you to create a list of technical terms and their definitions. In LaTeX you can automate this quite a lot. Definitions can be reused, entries are sorted, and, if you are producing a PDF document, then these can be linked from the words in the text to the relevant entry in the glossary.

Learning a new package is sometimes a challenge. The following suggestions techniques that I have used successfully.

- Do a proof of concept by creating a small working example.
- Make only small changes at a time and that what you have done works as you would expect.

How I made it work

i. First I read the documentation using `texdoc glossary`. I found it quite difficult to understand as there was a *lot* of information. Try it!

ii. Googled. When I searched for 'glossary package latex' the one of the first links was to `http://theoval.sys.uea.ac.uk/~nlct/latex/thesis/node25.html` which contained a useful summary. Note that this page was written by the author of the glossary package and so is likely to be authoritative.

Now to get started with your own glossary.

Cut and paste the following into a separate file called `glos.tex` starting here:

```
\documentclass[12pt,a4paper]{article}
\usepackage{glossary}
\makeglossary
\begin{document}
I'm going to write a paragraph
\glossary{name=paragraph,description=sentences
on a linked theme}
```

```
to demonstrate how glossaries \glossary{name=glossary,description=an
explanation of terms}
work. Glossaries usually appear at the end of documents, but can be
referred to at any time. In particular, I will describe how glossaries
work in  \LaTeX \glossary{name=latex,description=a mathematical typesetting  language}.
\printglossary
\end{document}
```

Compiling a `.tex` file that contains a glossary is a slightly more complicated procedure.

i. Build `glos.tex` with **PDFLaTeX**.

ii. Now in the command prompt window make sure you are in the same folder as your `glos.tex` file.

iii. Enter the command

```
makeindex -t glos.glg -o glos.gls -s glos.ist glos.glo
```

iv. Build `glos.tex` twice more and then look at the contents. You should see something like Figure 7.3.

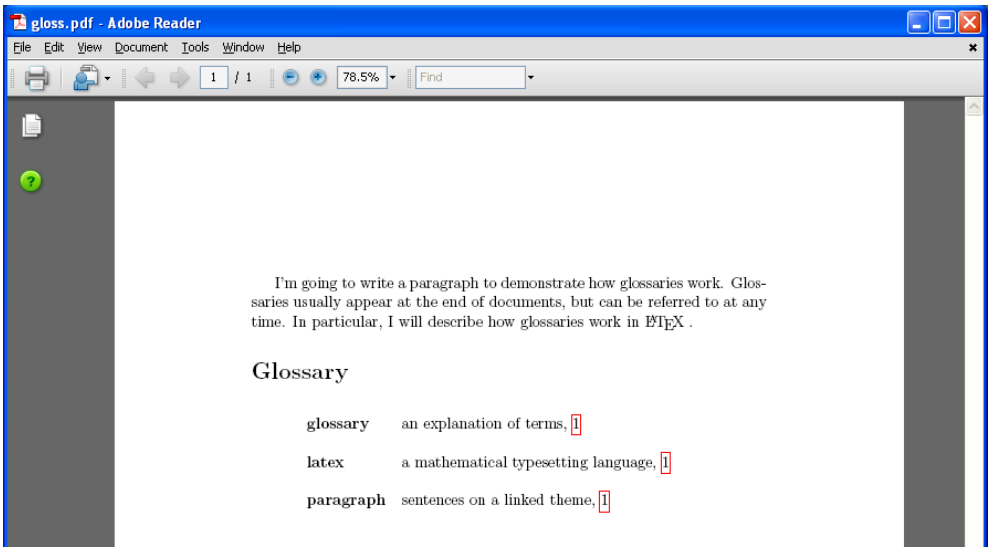

Figure 7.3: The first attempt at a glossary

It is possible to configure TeXnicCenter to build glossaries but this method will work whatever editor you are using.

Let's see how we can make this better. I would like the words in the text to be linked to the entry in the glossary. Reading the documentation it appears that `xglossary` would work as long as the `\hyperref` package is used. So we now have

```
\documentclass[12pt,a4paper]{article}
\usepackage{hyperref}
\usepackage{glossary}
\makeglossary
\begin{document}
I'm going to write a paragraph \xglossary{name=paragraph,
description=sentences on a linked theme}
to demonstrate how glossaries \xglossary{name=glossary,
description=an explanation of terms}
work. Glossaries usually appear at the end of
documents, but can be referred to at any time.
In particular, I will describe how glossaries work
```

```
in \LaTeX\ \xglossary{name=latex,description=a mathematical
typesetting language}. I will also describe amendments
\xglossary{name=amendments,description=a set of
changes or improvements} that can be made.
\printglossary
\end{document}
```

And my version:

I'm going to write a paragraph to demonstrate how glossaries work. Glossaries usually appear at the end of documents, but can be referred to at any time. In particular, I will describe how glossaries work in LaTeX for example. I will also describe amendments that can be made.

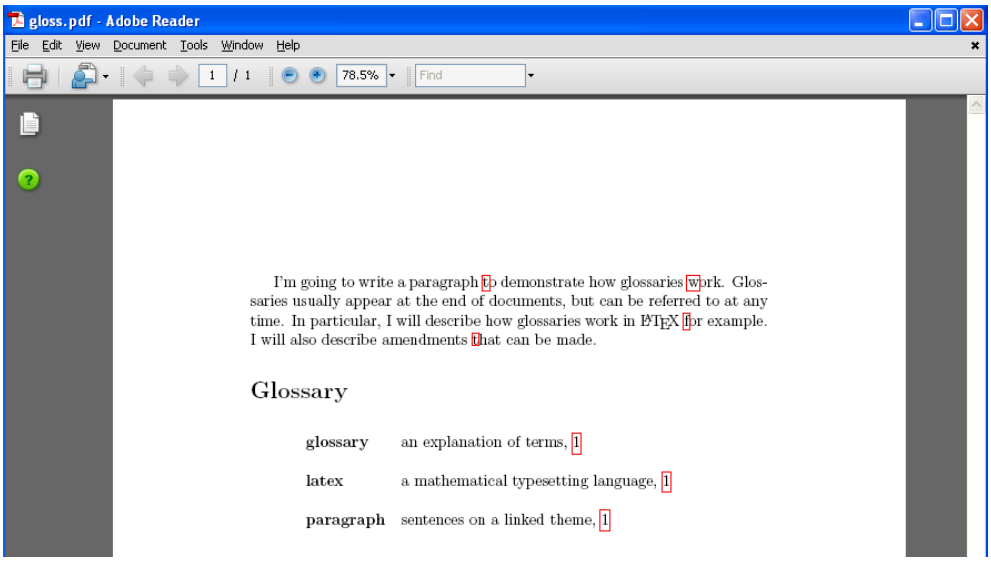

Figure 7.4: The second attempt at a glossary

I noticed that when I viewed the file displayed in Figure 7.4 that the links were to the first letter of the word *after* the glossary not to the word I intended. So I moved the words to after the glossary entry and surrounded them by {}. The paragraph now looks like this:

```
I'm going to write a \xglossary{name=paragraph,
description=sentences on a linked theme}{paragraph}
to demonstrate how \xglossary{name=glossary,
description=an explanation of terms}{glossaries}
work. Glossaries usually appear at the end of
documents, but can be referred to at any time.
In particular, I will describe how glossaries work
in \xglossary{name=latex,description=a mathematical
typesetting language}{\LaTeX}. I will also describe
\xglossary{name=amendments,description=a set of
changes or improvements}{amendments} that can be made.
```

And again:

I'm going to write a paragraph to demonstrate how glossaries work. Glossaries usually appear at the end of documents, but can be referred to at any time. In particular, I will describe how glossaries work in LaTeX. I will also describe amendments that can be made.

As you can see from Figure 7.5 the whole word links to its glossary entry.

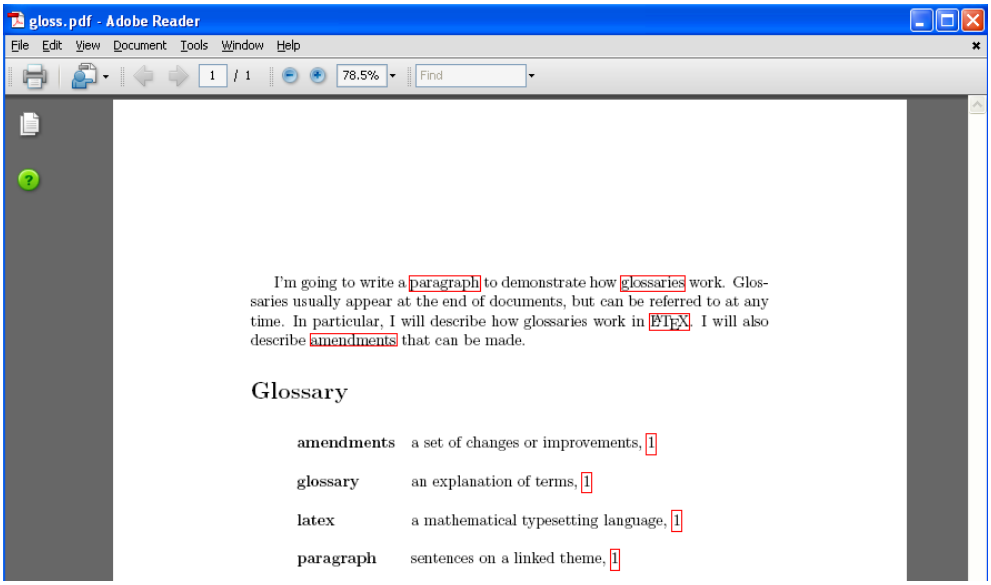

Figure 7.5: The third attempt at a glossary

What happens when a new entry is added? Run the command `makeindex -t glos.glg ...` again.

Finally what happens if I want to reuse entries. It makes sense to create definitions separately and then use a shortcut when linking text to the glossary. So my final version is:

```
\storeglosentry{glos:P}{name=paragraph,
description=sentences on a linked theme}
\storeglosentry{glos:G}{name=glossary,
description=an explanation of terms}
\storeglosentry{glos:L}{name=latex,
description=a mathematical typesetting
language}
\storeglosentry{glos:A}{name=amendments,
description=a set of changes or improvements}

I'm going to write a \useGlosentry{glos:P}
{paragraph} to demonstrate how \useGlosentry{glos:G}
{glossaries} work. \useGlosentry{glos:G}{Glossaries}
usually appear at the end of documents, but can be
referred to at any time. In particular, I will
describe how glossaries work in \useGlosentry{glos:L}
{\LaTeX}. I will also describe \useGlosentry{glos:A}
{amendments} that can be made.
```

And finally...

I'm going to write a paragraph to demonstrate how glossaries work. Glossaries usually appear at the end of documents, but can be referred to at any time. In particular, I will describe how glossaries work in LaTeX. I will also describe amendments that can be made.

Chapter 8

Debugging and troubleshooting

Have a look at the file `bad.tex`. Try to compile it. It should fail with several errors.

Your task is to fix the errors so that it looks like 8.1.

1 Introduction

The following *seven* characters are printed by typing a **backslash** in front of them: $ & # % _ { and }.

2 Main body

There are bullet lists and numbered lists.

- First bullet item.
- Second bullet item.
    1. First numbered item.
    2. Second numbered item.
    3. Third numbered item.
- Third bullet item.

These can be nested.

When typesetting maths remember you can use inline like this: $\tan(2\theta) = \frac{2\tan\theta}{1-\tan^2\theta}$. or display like this:

$$\tan(2\theta) = \frac{2\tan\theta}{1-\tan^2\theta}.$$

Figure 8.1: How the output should look

Chapter 9

Conclusion

This course has only touched on more advanced features of LaTeX.

Chapter 10

Answers

\include or \input?

See page 8. I think it makes more sense to use `\input{file.tex}` if you store the information from the preamble in a separate file. This information is always needed. It is also useful to do this so that it can be reused whenever you start a new LaTeX document.

Using arguments with new commands

See page 9.

```
\newcommand{\Uni}[1]{\textsf {\textbf {\color{blue} University of #1}}}
```

Customising existing commands

See page 10.

```
\renewcommand{\today}{\number\day\space \ifcase\month\or
  January\or February\or March\or April\or May\or June\or
  July\or August\or September\or October\or November\or December\fi
   \space\number\year}
```

If you have time...

```
\newcommand{\Uni}[2]{\textsf {\textbf {\color{#1} University of #2}}}
```

Creating a slide show

See page 12 **If you have time...**

There may well be other solutions but I found the following was needed to make the movie display

i. Add `\usepackage{multimedia}` to the preamble.
ii. Add `\movie[width=5cm, height=4cm]{Play me}{movie.avi}` to the frame.

Exploring packages

Reading this sort of documentation is not always easy, I find. There is often a lot of detail, particularly at the beginning and there is rarely a simple 'Getting started' section or instructions for frequently-used customisations. I must admit that I struggled to find the answer here and had to resort to Wikipedia! I added the following the preamble beneath `\usepackage{fancyhdr}`.

```
\fancyfoot{}
\lfoot{Susan Hutchinson}
\cfoot{}
\rfoot{\thepage}
```

This was quite easy to find out; the problem I had was that the page number was still centered on the first page of each chapter and no name appeared. This is perfectly acceptable, but I wanted my footer on *all* the pages. The solution was to change `\pagestyle{fancy}}` to `\pagestyle{fancyplain}}`

Debugging and troubleshooting

A possible solution is

```
\documentclass[a4paper,11pt]{article}
%Use article for short documents.
\usepackage[T1]{fontenc}
\usepackage{parskip}
\usepackage{latexsym,amsmath,amssymb}
\usepackage{times}
\begin{document}
\section{Introduction}
The following \emph{seven} characters are printed by
typing a \textbf{backslash} in front of them:  \$  \&  \#  \%  \_  \{  and  \}.
\section{Main body}
There are bullet lists and numbered lists.
\begin{itemize}
\item First bullet item.
\item Second bullet item.
\begin{enumerate}
\item First numbered item.
\item Second numbered item.
\item Third numbered item.
\end{enumerate}
\item Third bullet item.
\end{itemize}
These can be nested.

When typesetting maths remember you can use inline like this: $\tan (2\theta) =
\frac{2\tan\theta}{1-\tan^2\theta}.$ or display like this:

$$\tan (2\theta) = \frac{2\tan\theta}{1-\tan^2\theta}.$$
\end{document}
```

Glossary

amendments a set of changes or improvements, 18, 19

glossary an explanation of terms, 18, 19

latex a mathematical typesetting language, 18, 19

paragraph sentences on a linked theme, 18, 19

LaTeX Level 4

Further document preparation

Susan Hutchinson

Department of Statistics, University of Oxford.

Michaelmas 2014

Course outline and structure

Our LaTeX environment

- We will use MiKTeX and Texmaker in Windows.
- If you prefer to use the Command Prompt and Wordpad or Notepad then do so.

How does LaTeX work?

In order to use LaTeX two components are needed.

1. A LaTeX engine or distribution
   In Windows the most commonly used distribution is MiKTeX. It provides all the infrastructure for creating documents such as fonts, style files, compilation and previewing commands and much else. Another popular distribution is TeXLive which is widely used in Linux; Mac users generally use MacTeX.
2. An editor or IDE (Integrated Development Environment)
   This is used to edit, compile and preview LaTeX documents. There are many editors and IDEs available. Emacs is a popular editor which is available for both Windows and Linux; Lyx is available for both too and TeXworks runs on Windows, Linux and Macs. Several other platform-dependent editors are available such as Kile in Linux, TeXNicCenter, WinEDT and Winshell in Windows and Aquamacs for Macs. In Linux a simple text editor such as gedit can be a good option.

Developing LaTeX documents

When using Texmaker remember that

- LaTeX works on an edit → compile → view cycle.
- You must save your file after making changes
- You must recompile [PDFLaTeX] after any change and then [View PDF] to preview the file.
- Image files — plots, graphs, pictures — can cause confusion.
  - Documents with images in PostScript format (`.eps` or `.ps`) are compiled using [LaTeX].
  - Documents with images in PDF, PNG, or JPG format are compiled with [PDFLaTeX]. This is what we'll be using today.

A simple LaTeX file

```
\documentclass[a4paper,12pt]{report}
\begin{document}
\chapter{Introduction}
Hello there. This is the first paragraph.

Goodbye now. That's it.
\end{document}
```


A simple LaTeX file explained

The text of the document is surrounded by commands.

`\documentclass[a4paper,12pt{report}`

. . . This bit is called the preamble . . .

`\begin{document}`

. . . This bit is called the body . . .

`\end{document}`

Planning a longer document

When writing a longer document such as a thesis there are many advantages to breaking your document into separate files.

- Imposes a structure on the document as a whole.
- Allows you to focus on each part separately.
- Conditional compilation can speed up the compile → edit → preview cycle.

It is also possible to create a standard preamble which can be used whenever you start a new LaTeX document.

Structuring a longer document

When dividing up a longer document into separate files you need to do the following.

- Create a root file which is the file you use when compiling. This file contains markup which points to the files you want to include.
- When including files the `.tex` suffix is not needed.

When using Texmaker you will have many files open. Make sure you *always* compile your root file. Compiling any others will fail with many errors.

How to break up a long document

```
\documentclass[a4paper,12pt]{report}
\title{My long document}
\author{My Name}
\date
\begin{document}
\maketitle
\include{intro}
\include{theory}
\include{research}
\include{results}
\include{conclusion}
\end{document}
```

In this case the document is organised into five separate files called `intro.tex`, `theory.tex` and so on, each of which will contain the text of a single chapter.

Conditional compilation

There are many reasons why you may want to compile only one or two chapters. It is particularly useful if one chapter contains a lot of images and is slow to compile.

- Make sure that one compilation is done with all the chapters.
- Add the line

  `\includeonly{intro,theory}`

  in the preamble and recompile.
- Only the first two chapters will appear, but cross-referencing will be preserved, even to the chapters that have been left out.

Alternatively the `\excludeonly` command can be used to exclude one or more files.

Revision: labels and cross-references

LaTeX makes it easy to create cross-references to other parts of your document. You can create references to equations, figures, chapters, sections, pages, tables and so on. To create a reference to a chapter and page use the following:

- Add the following markup

  `\label{ch:intro}`

  at the beginning of the chapter. Labels are case-sensitive so INTRO is different from intro.
- Then when you want to make reference to the chapter add

  `In Chapter \ref{ch:intro} on \pageref{ch:intro}`

  when to reference the chapter and the page number.

Remember that you may need to compile your LaTeX file twice (at least) to get references right.

Exercises

Now do the following exercises.

- Creating, compiling and viewing a simple document using Texmaker
- Managing longer documents

Customising LaTeX

It is possible to define new commands and redefine existing ones. You may want to do this for several reasons.

- To save typing. If you keep typing the same markup it makes sense to define a command in the preamble and use the shortened version in the document.
- To keep a consistent style.

It is possible to create new commands that are shortcuts to existing commands.

```
\newcommand{\bi}{\begin{itemize}}
\newcommand{\ei}{\end{itemize}}
```


Creating your own commands

For example instead of writing

```
{\textrm {\slshape{Department of Statistics}}}
```

every time I want to write *Department of Statistics* I can define a new command in the preamble using

```
\newcommand{\DOS}{\textrm {\slshape Department
of Statistics}}
```

and use `\DOS` instead.

It is easy to change your mind later – the change would only need to be made in once in the preamble.

In slides the default font is sans–serif so in order to make my new command clearer I am using a Roman font. In documents it is the other way round so use `\textsf` instead.

Defining a new command with an argument

The syntax of the command to define a new command is

```
\newcommand{cmd}[n-args][default]{text}
```

where

- cmd is the name you decide to use for the new command.
- n-args [optional] is the number of arguments your new command will have.
- default [optional] what to use as a default argument.
- text what LaTeX should do with when this command is used.

To define a command called *keyword* which typesets a word or phrase in bold using a typewriter font and coloured blue add to the preamble

```
\newcommand{\keyword}[1]{\texttt{\slshape\color{blue}#1}}
```

So to typeset the word "section" as a keyword use `\keyword{section}` which would appear as ***`section`***.

Customising an existing command

Using `\renewcommand` the behaviour of existing commands can be changed. For example to change the page with your table of contents to say "Table of Contents" rather than "Contents" you would use

```
\renewcommand*{\contentsname}{Table of Contents}
```

There are many predefined names which can be changed.

```
\contentsname
\listfigurename
\listtablename
\figurename
\chaptername
```

Note that some names will only apply to some documentclasses. For example there is no Chapter command when using the `article` class. The use of an asterisk (star) in a `\renewcommand` indicates that the command is *short* and unlikely to include a paragraph break.

Advice on customising commands

Customisation needs to be done with care. LaTeX imposes good typesetting rules on your document so you need to make sure your changes are necessary.

- LaTeX does not allow you to create new commands with the same name as existing commands.
- Check that there isn't a package which will achieve the effect you are looking for, before trying to make extensive customisations to an existing command.
- If you do end up creating many customisations your preamble may become rather long; it is relatively simple to bundle them all up into your own package which can then be invoked with `\usepackage{mypackage}`.

Why use a LaTeX slide-making environment?

. . . when there's Microsoft Powerpoint?

- It is straightforward to include mathematics.
- It is possible to link to bibliographies.
- Customisation is possible, so that the environment can be tailored to suit your needs.

So the same considerations apply. If you have chosen to use LaTeX to write your thesis then you will probably need to be able to create presentations in LaTeX too.

Creating slideshows

There are several slide-making document classes. Originally there were several classes that were suited to creating transparencies and foils.

SliTeX A separate program, written by Leslie Lamport for creating transparencies.

slides Early LaTeX slide-making document class. It is not widely used now as it lacks many more complex features. It was good for creating transparencies; less so for online presentations.

seminar and foils Originally developed to produce acetate foils, but can produce output suitable for an overhead projector. Not much used.

Powerpoint changed everything. The expectation was that a presentation, as well as containing useful information would also be colourful, include dynamic effects such as animations and pausing between bullet points.

Current slide-making classes

Several new slide-making environments were produced in response.

prosper Based on seminar and includes the ability to produce dynamic effects. Now superceded largely by powerdot.

beamer Relatively powerful and easy to learn; creating dynamic effects is relatively straightforward.

talk Again, easy to learn. It doesn't impose a particular slide-style on you.

We will be using Beamer today. Early distributions of LaTeX did not always include Beamer; but if you are using an up-to-date version then it should be available.

Beamer will produce output in PDF format which makes it very portable. PDF format presentations will "just work" on most systems.

What Beamer provides

Beamer extensions

There are several additional environments and commands that are specific to Beamer.

- New environments include *block*, *column* and *animate*.
- There are also several mathematical environments such as *theorem*, *proof* and *definitions*.
- Transitions, pauses and overlays are easily managed.
- That was a pause.

beamercolorbox allows you to change both the background and foreground colour of a part of a slide.

A simple beamer file

```
\documentclass[pdf]{beamer}
\usetheme{Warsaw}
\title{My first slideshow}
\subtitle{I hope you like it}
\author{Susan Hutchinson}
\institute{University of Oxford}
\date
\begin{document}
\maketitle
\tableofcontents
\section{Introduction}
\begin{frame}
```

The contents of the slide go here.

```
\end{frame}
\end{document}
```


The components of a slide

1. a headline and a footline
2. a left and right sidebar
3. navigation bars
4. navigation symbols
5. a logo
6. a frametitle
7. a background
8. some frame contents

Not all slides have all these components. The first three are usually set up by the theme you choose. The contents are your problem!

Beamer themes and colours

The appearance of your slides depends on the theme you have chosen. There is a smaller range than with powerpoint but there is much scope for customisation. To change the theme and colour use

```
\usetheme{PaloAlto}
\usecolortheme{albatross}
```

Some popular themes are

AnnArbor	Berkeley	Berlin	Boadilla
CambridgeUS	Copenhagen	Darmstadt	Frankfurt
Hanover	Luebeck	Malmoe	PaloAlto
Pittsburgh	Rochester	Singapore	Warsaw

Some popular colours are

albatross	beaver	beetle	crane	dolphin	dove
fly	orchid	rose	seagull	seahorse	wolverine

I am using the `Luebeck` theme and `rose` .

The layout of a slide

Each slide has the format

```
\begin{frame}
\frametitle{}
    The contents of the slide go here.
\end{frame}
```

The contents of the slide can include LaTeX commands, pictures, tables and so on.

Beamer restrictions

- The depths of *itemize*,*enumerate* and *description* environments are limited.
- Pictures and figures need careful handling.
- Using *bibtex* is rather fiddly.

The title page

Add the following to the preamble

```
\title[Short title]{My long title}
\subtitle[Short subtitle]{My long subtitle}
\author{My Name}
\date{November 2007}
\institute{My University}
```

and then include

```
\begin{frame}
\maketitle
\end{frame}
```

after `\begin{document}`. A short version of the text has been included between `[` and `]` which will appear at the foot of each slide.

Effects can be included such a pauses and overlays. For example to pause between items like this:

- Mount Everest grows by 1cm a year
- A new planet is discovered every day.

use

```
\begin{itemize}
\item Mount Everest grows by 1cm a year
\pause
\item A new planet is discovered every day.
\end{itemize}
\pause
```

Creating overlays

Overlays allow you to determine in what order items appear. For example

Theorem

There is no largest prime number.

Proof.

1. Suppose p where the largest prime number.
2. Let q be the product of the first p numbers.
3. Then $q + 1$ is not divisible by any of them.
4. Thus $q + 1$ is also prime and greater than p. □

Proved using *reduction ad absurdum*.

The markup for overlays

```
\begin{theorem}
There is no largest prime number.
\end{theorem}
\begin{proof}
\begin{enumerate}
\item<1-> Suppose $p$ where the largest prime number.
\item<2-> Let $q$ be the product of the first $p$ numbers.
\item<3-> Then $q + 1$ is not divisible by any of them.
\item<1-> Thus $q + 1$ is also prime and greater than
 $p$. \qedhere
\end{enumerate}
\end{proof}
\uncover<4->{Proved using \textit{reduction ad absurdum}.}
```

Note the use of <1->, <2-> to determine the order in which information is revealed.

Adding some structure

The `\section` and `\subsection` commands are used to add structure to the slides. These are used outside frames. They can contain a long and short version. The long version appears in the table of contents, the short version in the header line.

```
\section[Slide creation]{Creating slides using Beamer}
```

Add a slide that contains

```
\begin{frame}
\frametitle{Outline}
\tableofcontents
\end{frame}
```

and a slide which includes all the section and subsections headings will be generated.
Finally navigation symbols can be added which allow you to find your way around the presentation when it is being given.

Exercises

Now do the following exercises.

- Creating and customising commands
- Make your own slide show

What are packages

Packages are used to alter or add to basic LaTeX behaviour. For example to change the way paragraphs are separated add `\usepackage{parskip}` to the preamble.

```
\documentclass{article}
\usepackage{parskip}
\begin{document}
\section{Introduction}
Hello there. This is the first paragraph.

Goodbye now. That's it.
\end{document}
```


More about packages

Finding and configuring packages to do what you want is a key skill for LaTeX users.

- Packages extend existing functions.
- Packages add extra functions.
- There are hundreds of packages, many of which will be installed with your LaTeX distribution.

Some commonly-used packages

amsmath,amssymb	Additional mathematical characters
fancyhdr	Extends headers and footers on the page
graphicx	More configuratable picture environment
longtable	Allows tables to extend over more than one page
lscape	Change the orientation of a page
natbib	Add a bibliography
tocloft	Changes Table of Contents format

Some more commonly used packages

babel	Allows you to choose the language of key words.
color, xcolor	Change the colour of text and other features.
fancyvrb	Enhances the verbatim environment. Useful if you need to make text appear exactly as it is typed.
glossaries	Create a glossary of key words.
hyperref	Make your references, tables of contents, lists of figures and tables links.

Learning a new package

Once you have found the package you want, you will need to understand how it works.

- Read the documentation. Most packages have a guide which can be read with the *texdoc* command if you are using a Unix/Linux system.
- Use Google.
- Create a small working example to help you become familiar with the features.

Solving your own problems

These slides and exercises are designed to get you started. At some point you will hit a problem (I do all the time) which you can't solve immediately.

- Move `\end{document}` further up the file. This may isolate the problem.
- If the error reports a line number make sure you look for errors before that.
- Use Google to search for the error. Or look at a guide.
- Create a minimal example. This helps you narrow the problem down.
- Have a look at the TeX– LaTeX site on StackExchange.

As an exercise, I have included a broken `.tex` file for you to fix.

Exercises

Now do the following exercise.

- Exploring packages.
- Fix this file!

If you have problems check that you have used the suggestions in the previous slide.

Useful links

`http://www.ctan.org` Download LaTeX and other resources.

`http://www.tug.org` TeX Users group.

`http://www.dickimaw-books.com/latexresources.html` Lots of useful advice and teaching material.

`http://stackexchange.com/` Useful and reliable question and answer forum.

`detexify.kirelabs.org/classify.html` Find the markup for mathematical symbols.

The last slide

Good luck!

I hope you will all now write beautiful documents.